G-Adapter: Towards Structure-Aware Parameter-Efficient Transfer Learning for Graph Transformer Networks

Anchun Gui, Jinqiang Ye, and Han Xiao*

Department of Artificial Intelligence
School of Informatics, Xiamen University
{anchungui, jinqiangye}@stu.xmu.edu.cn, bookman@xmu.edu.cn

Abstract

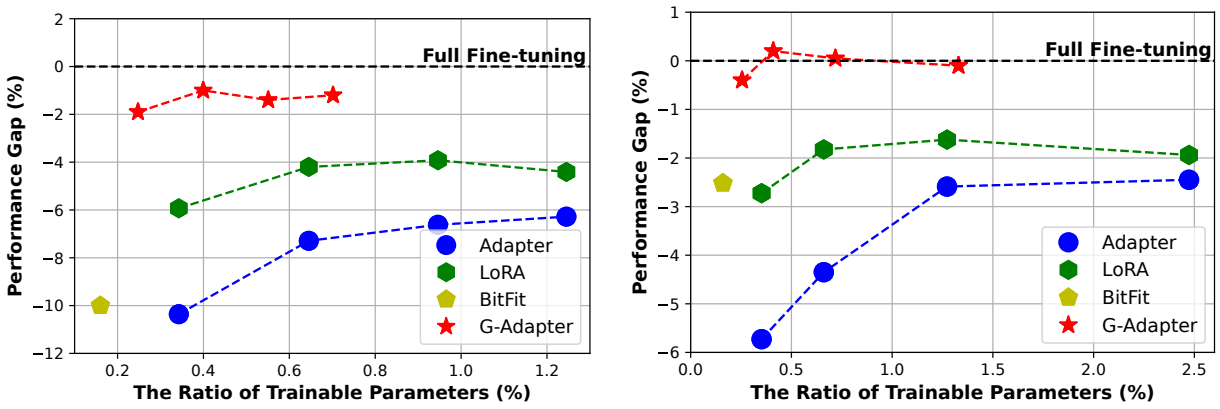
It has become a popular paradigm to transfer the knowledge of large-scale pre-trained models to various downstream tasks via fine-tuning the entire model parameters. However, with the growth of model scale and the rising number of downstream tasks, this paradigm inevitably meets the challenges in terms of computation consumption and memory footprint issues. Recently, Parameter-Efficient Fine-Tuning (PEFT) (e.g., Adapter, LoRA, BitFit) shows a promising paradigm to alleviate these concerns by updating only a portion of parameters. Despite these PEFTs having demonstrated satisfactory performance in natural language processing, it remains under-explored for the question: whether these techniques could be transferred to graph-based tasks with Graph Transformer Networks (GTNs)? Therefore, in this paper, we fill this gap by providing extensive benchmarks with traditional PEFTs on a range of graph-based downstream tasks. Our empirical study shows that it is sub-optimal to directly transfer existing PEFTs to graph-based tasks due to the issue of ***feature distribution shift***. To address this issue, we propose a novel structure-aware PEFT approach, named G-Adapter, which leverages graph convolution operation to introduce graph structure information (e.g., graph adjacency matrix) as an inductive bias to guide the updating process. Further, we propose Bregman proximal point optimization to alleviate feature distribution shift by preventing the model from aggressive update. Extensive experiments demonstrate that G-Adapter obtains state-of-the-art performance compared to counterparts on nine graph benchmark datasets based on diverse pre-trained GTNs, and delivers tremendous memory footprint efficiency compared to the conventional paradigm.

Introduction

Pre-training then fine-tuning has become a prevalent training paradigm, with the remarkable success of large-scale pre-trained models in Natural Language Processing (NLP) (Devlin et al. 2019; Liu et al. 2019; Raffel et al. 2020; Lewis et al. 2020; Brown et al. 2020). Recently, more researchers are striving to apply this paradigm to graph-based tasks with Graph Transformer Networks (GTNs) (Maziarka et al. 2020; Ying et al. 2021; Mialon et al. 2021; Zhao et al. 2021; Kim et al. 2022; Kreuzer et al. 2022; Bo et al. 2023; Jin et al. 2023; Yuan et al. 2023). For instance, based on multi-layer Transformer encoders (Vaswani et al. 2017), Graphormer (Ying et al. 2021) first performs the well-designed unsupervised tasks on large-scale molecular datasets, and then fine-tunes the entire pre-trained parameters of the model on downstream molecular tasks of interest, which is also known as *full fine-tuning*. However, full fine-tuning poses several issues in practice: (i) Given that the labels of graph data from some domains (e.g., chemistry, biology) are inaccessible without the expertise and labor-heavy annotations (Xia et al. 2022), it is common that there are insufficient labeled samples in downstream tasks of interest. Therefore, full fine-tuning would incur serious over-fitting and catastrophic forgetting issues (Houlsby et al. 2019; Wang et al. 2021). (ii) When handling multiple diverse downstream tasks, full fine-tuning has to duplicate a modified copy of all parameters per task, which hinders the flexibility and applicability of large-scale models, especially in scenarios with constrained storage resources (e.g., mobile detection devices).

Recently, Parameter-Efficient Fine-Tuning (PEFT), as an alternative to full fine-tuning, has been proposed and widely investigated in NLP (Houlsby et al. 2019; Ben-Zaken, Goldberg, and Ravfogel 2022; Ding et al. 2022; He et al. 2022; Hu et al. 2022). PEFT aims to achieve competitive performance with full fine-tuning while consuming computation and storage resources as few as possible. Instead of updating the entire parameters during the fine-tuning phase, PEFT only updates a small fraction of parameters within the original model or additionally introduced modules, while freezing the remaining parameters. For example, Adapter (Houlsby et al. 2019) inserts two compact modules in each encoder of Transformer, while BitFit (Ben-Zaken, Goldberg, and Ravfogel 2022) only updates the bias terms in the model parameters, as shown in Fig. 3. Despite the remarkable achievements of traditional PEFTs in natural language understanding tasks, the question is still under-explored whether these PEFTs from the language domain are feasible for various GTNs under graph-based tasks, given that the intrinsic discrepancy between graph and text modalities (e.g., the graph has rich structure information). Therefore, in this paper, we shall fill this gap by answering the following questions: *Can PEFTs from the language domain be transferred directly to graph-based tasks? If not, how to design a graph-specific PEFT method?*

*Corresponding author.



(a) On large-scale datasets. (b) On small-scale datasets.

Figure 1: The comparison between PEFTs (Adapter, LoRA, BitFit, and G-Adapter) and full fine-tuning on large- and small-scale datasets, respectively. Here, we can observe that there is a significant performance gap between traditional PEFTs (Adapter, LoRA, BitFit) and full fine-tuning, while our proposed G-Adapter could obtain comparable results with full fine-tuning on large-scale datasets (and even better performance on small-scale datasets). Refer to pilot experiments for more settings.

To start with, we comprehensively examine the performance of mainstream PEFTs, such as Adapter (Houlsby et al. 2019), LoRA (Hu et al. 2022), and BitFit (Ben-Zaken, Goldberg, and Ravfogel 2022), on popular molecular graph datasets based on two pre-trained GTNs (i.e., Graphormer (Ying et al. 2021), MAT (Maziarka et al. 2020)). The overall comparison is shown in Fig. 1, in which we unfortunately observe a significant gap between traditional PEFTs and full fine-tuning, especially on large-scale datasets. Our exploration further reveals *feature distribution shift* issue in traditional PEFTs due to the absence of graph structure information in the fine-tuning process (see Fig. 2 and pilot experiments). To alleviate these concerns, we propose a novel structure-aware PEFT method, called G-Adapter, which employs graph convolution operation to explicitly introduce graph structure information as the inductive bias to guide the updating process. Meanwhile, we apply low-rank decomposition to the learnable weights, which makes G-Adapter highly lightweight. Besides, we propose Bregman proximal point optimization to further ease the feature distribution shift by preventing the model from aggressive update.

To verify the effectiveness of our approach, we conduct extensive experiments on a variety of graph-based downstream tasks based on diverse pre-trained GTNs. The results demonstrate that our proposed G-Adapter can effectively address the feature distribution shift issue and significantly enhance the performance. Specifically, (i) G-Adapter obtains state-of-the-art performance than baselines on both large- and small-scale datasets. Even compared to full fine-tuning, G-Adapter could achieve competitive (or superior) results with fewer trainable parameters. For example, full fine-tuning achieves 0.804 AUC with 100% trainable parameters on MolHIV, while G-Adapter obtains 0.790 AUC with only 0.24% trainable parameters. (ii) G-Adapter enjoys remarkable advantages over full fine-tuning in terms of storage footprint. For instance, full fine-tuning stores 161MB checkpoint per task, while G-Adapter merely requests 0.4MB checkpoint for each task. Additionally, the introduced G-Adapter modules barely degrade the training and inference efficiency, and extensive ablation experiments also confirm the rationality of each component in our methodology.

To summarize, our contributions are as follows:

- To the best of our knowledge, this is the first work formally to investigate the parameter-efficient fine-tuning of graph-based tasks and models. And, we benchmark several widely used PEFTs from the language domain on a range of graph-based downstream tasks.
- We demonstrate a phenomenon of feature distribution shift when directly applying existing PEFTs to graph-based tasks. Further, our study empirically shows that the graph structure information and Bregman proximal point optimization strategy could alleviate this concern well.
- We propose a structure-aware parameter-efficient fine-tuning method (G-Adapter) for adapting pre-trained GTNs to various graph-based downstream tasks, in which G-Adapter introduces graph structure information as the inductive bias to guide the updating process.
- Extensive experiments demonstrate that G-Adapter outperforms counterparts by a significant margin. Furthermore, compared to full fine-tuning, our method yields tremendous memory footprint benefits almost without sacrificing the efficiency of training and inference.

Related Work

Graph Transformer Networks. Transformer (Vaswani et al. 2017) has demonstrated remarkable advantages in NLP (Devlin et al. 2019; Brown et al. 2020), which spurs extensive research on transferring Transformer to graph representation learning (Min et al. 2022; Xia et al. 2022). Compared to the simple sequential relationship between tokens in the text, the relationship between edges in the graph

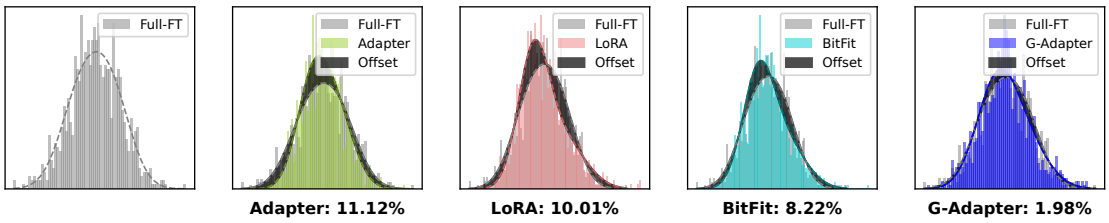


Figure 2: The illustration of feature distribution shift, where Full-FT denotes full fine-tuning. For the identical input, the feature distribution of traditional PEFTs (Adapter, LoRA, and BitFit) has a significant offset (dark region) compared to full fine-tuning. In contrast, our proposed G-Adapter has a highly similar behavior to full fine-tuning. Here, Jensen-Shannon divergence is utilized to measure the discrepancy between two feature distributions. Refer to pilot experiments for more detailed discussions.

could be more complex and essential (Min et al. 2022; Xia et al. 2022). Therefore, substantial works focus on modeling graph structures (Maziarka et al. 2020; Khoo et al. 2020; Ying et al. 2021; Zhao et al. 2021; Kreuzer et al. 2022). For example, Graphormer (Ying et al. 2021) leverages the centrality and spatial encoding as the graph structural signal, and MAT (Maziarka et al. 2020) augments the attention mechanism in Transformer using inter-atomic distances and molecular graph structure. Kreuzer et al. (2022) propose the learnable structural encoding via Laplacian spectrum, which can learn the position of each node in the graph. Moreover, Zhao et al. (2021) proposes a proximity-enhanced multi-head attention to capture the multi-hop graph structure, and Khoo et al. (2020) design a structure-aware self-attention for modeling the tree-structured graphs. Additionally, Min et al. (2022) systematically investigate the effectiveness and application of Transformers in the graph domain.

Parameter-Efficient Transfer Learning. Parameter-Efficient Fine-Tuning (PEFT), as an alternative to full fine-tuning, is receiving extensive attention in research community (Ding et al. 2022; He et al. 2022; Yu et al. 2022). A major advantage of PEFT is that it significantly reduces the demand on GPU-memory and storage footprint while maintaining comparable performance as full fine-tuning. This property is especially beneficial for training large pre-trained model under limited resources. To this end, a series of methods are proposed. For example, Adapter (Houlsby et al. 2019) inserts two lightweight blocks into the Multi-Head Attention (MHA) and Feed-Forward Networks (FFN) layers in Transformer, Compacter/Compacter++ (Mahabadi, Henderson, and Ruder 2021) introduce Kronecker product and weights sharing tricks to further reduce the proportion of trainable parameters. Based on the hypothesis of "low intrinsic rank" (Aghajanyan, Gupta, and Zettlemoyer 2021), LoRA (Hu et al. 2022) tunes two low-rank matrices to approximate the updating of query and value weights in MHA. Besides, instead of introducing extra parameters, BitFit (Ben-Zaken, Goldberg, and Ravfogel 2022) only fine-tunes the bias terms of the model. More PEFT works are included here (Li and Liang 2021; Rücklé et al. 2021; Liu et al. 2022; Jie and Deng 2022; Zhang et al. 2023; Chen et al. 2023). Finally, He et al. (2022) provide a unified view of existing PEFTs, and more detailed descriptions of PEFT methods are discussed in this survey (Ding et al. 2022).

Methodology

Pilot Experiments

To answer the first question: *Can PEFTs from the language domain be transferred directly to graph-based tasks?* We evaluate the performance of three traditional PEFTs (Adapter, LoRA, BitFit) on large- and small-scale graph-based downstream tasks, respectively. To be specific, on two large-scale MolHIV (41K) and MolPCBA (437K) (Hu et al. 2021) datasets, we calculate the performance gap between PEFTs and full fine-tuning according to their average results on these datasets, based on the pre-trained Graphormer model (Ying et al. 2021). Similar computations are conducted on seven small-scale datasets (0.6∼2.4K): FreeSolv, ESOL, BBBP, Estrogen-α, Estrogen-β, $\text{MetStab}_{\text{low}}$, and $\text{MetStab}_{\text{high}}$ (Gaulton et al. 2012; Podlewska and Kafel 2018; Wu et al. 2018), based on another pre-trained MAT model (Maziarka et al. 2020). It can be observed in Fig. 1 that traditional PEFTs lag far behind that of full fine-tuning on graph-based tasks, especially on large-scale datasets, across a range of the ratio of trainable parameters.

To shed light on why there is such a significant performance gap between traditional PEFTs and full fine-tuning, we visualize the feature distribution of different methods and compare their discrepancy using Jensen-Shannon divergence[1]. Intuitively, considering that full fine-tuning updates all model parameters and obtains satisfactory performance on downstream tasks, it can be regarded as an "upper bound" of PEFTs and thus its corresponding feature encoding for the graph-based input is relatively optimal. Therefore, we hypothesize that a good PEFT should have similar feature distribution with full fine-tuning. However, we can observe in Fig. 2 that the feature distributions encoded by traditional PEFTs are shifted compared to full fine-tuning, which here is called ***feature distribution shift***. To further explore the reason underlying this phenomenon, we revisit the relationship between GTNs and vanilla Transformers. Although they share many commonalities (e.g., the model architecture), there are significant discrepancies in terms of the modeling of input. For example, only the token and po-

[1]For the computation of Jensen-Shannon divergence, supposing we have two feature distributions P and Q, then $\text{JS}(\text{P}||\text{Q}) = \frac{1}{2}\text{KL}(\text{P}||\frac{\text{P}+\text{Q}}{2}) + \frac{1}{2}\text{KL}(\text{Q}||\frac{\text{P}+\text{Q}}{2})$, where $\text{JS} \in [0, 1]$ and the smaller JS, the more similar the two distributions are.

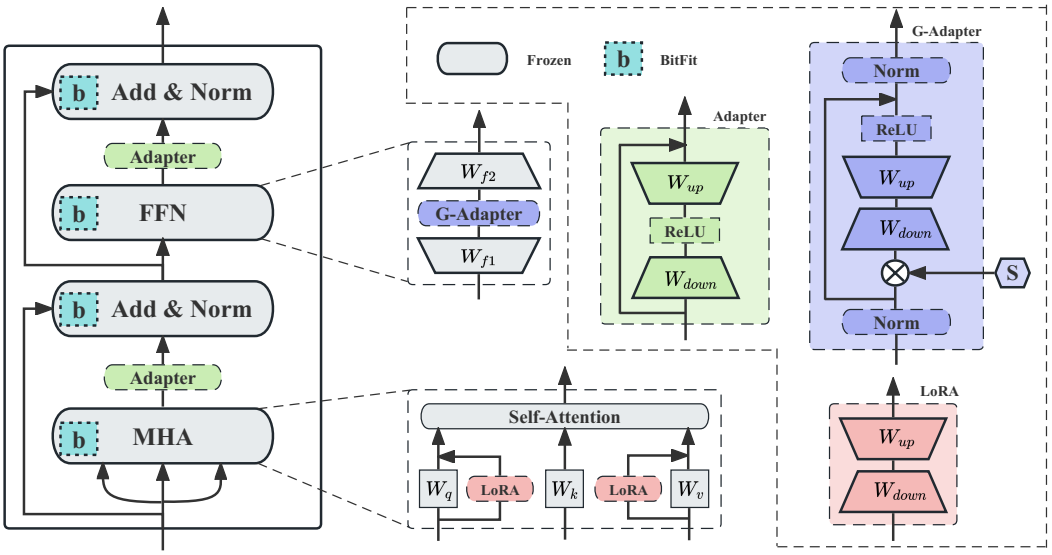


Figure 3: An overview of existing popular PEFTs (Adapter, LoRA, and BitFit) and our proposed G-Adapter. In the left subfigure, we demonstrate the insertion positions of various PEFT blocks in a standard encoder of Transformer. In the right subfigure, we depict the detailed architecture of each PEFT (more rigorous mathematical descriptions are in Appendix A.1). Each color represents an approach, and the grey blocks are frozen during the fine-tuning process. Our proposed G-Adapter is marked and demonstrated in purple, where S indicates the introduced graph structure information (e.g., graph adjacency matrix).

sition embeddings are utilized within vanilla Transformers, while GTNs require extracting diverse graph structures from the graph-based input to capture the complex edge connections in the graph (Maziarka et al. 2020; Zhang et al. 2020; Ying et al. 2021; Zhao et al. 2021). In addition, recent research also demonstrates the significant effectiveness of graph structure in graph representation learning (Kreuzer et al. 2022; Diao and Loynd 2023). Motivated by these observations, in this paper, we attempt to introduce graph structure information as the inductive bias to alleviate the feature distribution shift issue.

Structure-Aware Parameter-Efficient Fine-Tuning

The Design of Parameter-Efficient Module. Inspired by Graph Convolutional Networks (GCN) (Kipf and Welling 2017; Lin, Wang, and Liu 2021), which can model both graph structure and node representation simultaneously, we leverage this operation to explicitly introduce graph structure into the model. Here, we give the following definition:

$$X' = \text{GraphConv}(S, X; W) = \sigma(SXW) \qquad (1)$$

where $X, X' \in \mathbb{R}^{n\times d}$ (n: the sequence length, d: the hidden representation dimension) refer to the input, output of this module, respectively. $S \in \mathbb{R}^{n\times n}$ indicates the introduced graph structure information (e.g., the adjacency matrix of graph), $W \in \mathbb{R}^{d\times d}$ is a learnable weight, and $\sigma(\cdot)$ indicates the nonlinear activation function. Further, following the lightweight principle (Houlsby et al. 2019), we decompose W into two low-rank matrices to reduce the number of learnable parameters, i.e., $W = W_{down}W_{up}$, where $W_{down} \in \mathbb{R}^{d\times r}, W_{up} \in \mathbb{R}^{r\times d}$ and r is called the bottleneck size. Moreover, to stabilize the training process, we insert two LayerNorm layers (Ba, Kiros, and Hinton 2016) before and after $\text{GraphConv}(\cdot)$, respectively, as depicted in Fig. 3. Overall, the pipeline of our G-Adapter module is as follows:

$$X' = \text{LN}(X), X'' = \text{LN}\big(X' + \sigma(SX'W_{down}W_{up})\big) \qquad (2)$$

where $\text{LN}(\cdot)$ represents the LayerNorm layer. In addition, thanks to the modular and lightweight properties, G-Adapter can be seamlessly integrated into diverse GTNs.

The Selection of Graph Structure Information. To start with, we consider the adjacency matrix (with self-connections) of the graph: $S_1 = A + I_n$, where $A, I_n \in \mathbb{R}^{n\times n}$ refer to the adjacency matrix and the identity matrix, respectively. Then, following the setting from Kipf and Welling (2017), we introduce the degree matrix of nodes to normalize the adjacency matrix: $S_2 = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = S_1$, $\tilde{D}$ is the diagonal matrix, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. In addition, we propose a distance-based graph structure information: $S_3 = [\text{dis}(v_i, v_j)]_{n\times n}$, where $\text{dis}(v_i, v_j)$ refers to the distance of the shortest path (or the inter-atomic distance in the molecular graph) between two nodes v_i and v_j. Last, we combine the adjacency and distance to construct a hybrid structure information: $S_4 = \alpha \cdot \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} + \beta \cdot [\text{dis}(v_i, v_j)]_{n\times n}$, where α, β are scalar hyper-parameters to balance the impacts of the adjacency and distance terms. We evaluate the proposed graph structure information (S_1, S_2, S_3, S_4) on a range of graph-based tasks, and the detailed comparisons are presented in Table 1 and 2.

Bregman Proximal Point Optimization

It is expected that the feature distribution encoded by PEFT should be aligned with full fine-tuning as much as possible,

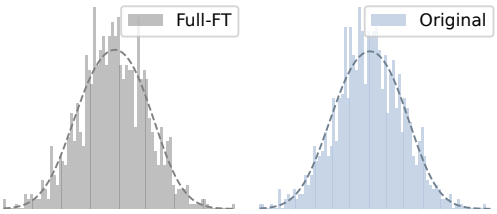


Figure 4: Comparison of the feature distribution between full fine-tuning and the original model parameters, where Jensen-Shannon divergence is 0.27%.

as discussed in pilot experiments. However, the feature distribution of full fine-tuning is unavailable during the training process of PEFT. Interestingly, we observe that the feature distribution encoded by the original model parameters has a high similarity with full fine-tuning, as shown in Fig. 4. It indicates that full fine-tuning only slightly modulates the value of model parameters but does not change the ability of the model to encode features. Therefore, to maintain consistency with the feature distribution of the original parameters, we propose Bregman proximal point optimization strategy (Jiang et al. 2020) to prevent the model from aggressive update. Specifically, for the pre-trained model $f(\cdot;\theta)$ with trainable parameters θ, at the $(t+1)$-th iteration, we have

$$\theta_{t+1} = \arg\min_{\theta} \ (1-\mu)\cdot\mathcal{L}_{\text{vanilla}}(\theta) + \mu\cdot\mathcal{L}_{\text{bregman}}(\theta,\theta_t) \tag{3}$$

where $\mu > 0$ is a hyper-parameter, $\mathcal{L}_{\text{vanilla}}$ is a common classification or regression loss function, and $\mathcal{L}_{\text{bregman}}$ is the Bregman divergence defined as:

$$\mathcal{L}_{\text{bregman}}(\theta,\theta_t) = \mathbb{E}_{x\sim\mathcal{D}}\Big[\ell\big(f(x;\theta), f(x;\theta_t)\big)\Big] \tag{4}$$

where the input x is derived from the training set $\mathcal{D}$, and here we leverage the symmetric KL-divergence, i.e., $\ell(p,q) = \mathrm{KL}(p||q)+\mathrm{KL}(q||p)$. Intuitively, $\mathcal{L}_{\text{bregman}}$ serves as a regularizer and prevents θ_{t+1} from deviating too much from the previous iteration θ_t, therefore can effectively retain the capacity of encoding feature in the pre-trained model $f(\cdot;\theta)$.

Experiments

Setup

Datasets & Evaluation Protocols. Our experiments are conducted on nine datasets: MolHIV, MolPCBA, FreeSolv, ESOL, BBBP, Estrogen-α, Estrogen-β, $\text{MetStab}_{\text{low}}$, and $\text{MetStab}_{\text{high}}$ (Gaulton et al. 2012; Hu et al. 2021; Podlewska and Kafel 2018; Wu et al. 2018). Here, MolHIV and MolPCBA are two large-scale molecular property prediction datasets, and the others are small-scale molecular datasets. Following the previous settings (Maziarka et al. 2020; Ying et al. 2021), MolPCBA is evaluated by Accuracy Precision (AP), FreeSolv and ESOL are evaluated by RMSE, and the others are evaluated by AUC.

Pre-trained Models & Baselines. Graphormer (Ying et al. 2021) and MAT (Maziarka et al. 2020) are leveraged as our backbone models, where Graphormer/MAT has $12/8$ encoders of Transformer. For the baselines, we include full fine-tuning (denoted Full FT) and six popular traditional PEFTs: Adapter (Houlsby et al. 2019), LoRA (Hu et al. 2022), BitFit (Ben-Zaken, Goldberg, and Ravfogel 2022), Hyperformer (denoted Hyper.) (Karimi Mahabadi et al. 2021), Compacter (denoted Comp.) (Mahabadi, Henderson, and Ruder 2021), and MAM (He et al. 2022).

Method	MolHIV		MolPCBA	
	Ratio (γ)	AUC ($\uparrow$)	Ratio (γ)	AP ($\uparrow$)
Full FT	100%	0.804	100%	0.272
Adapter	1.24%	0.743	4.69%	0.235
Hyper.	1.13%	0.740	4.37%	0.246
Comp.	0.64%	0.752	3.42%	0.230
MAM	0.57%	0.758	2.66%	0.251
LoRA	0.34%	0.763	2.42%	0.246
BitFit	0.16%	0.709	0.16%	0.184
Ours (S_1)	0.24%	**0.790**	1.89%	**0.269**
Ours (S_2)	0.24%	0.788	1.89%	0.264
Ours (S_3)	0.24%	0.778	1.89%	0.250
Ours (S_4)	0.24%	0.781	1.89%	0.262

Table 1: Comparison on two large-scale datasets MolHIV and MolPCBA, where bold indicates the best result.

Implementation. To begin with, we initialize our backbone models with the official released pre-trained checkpoints, note that the introduced modules are randomly initialized. Then, Adam optimizer (Kingma and Ba 2015) is used to fine-tune our models, and we set fair hyperparametric search budgets for various PEFTs.

Main Results

We report the experimental results on MolHIV and MolPCBA based on the pre-trained Graphormer in Tab. 1, and more results on small-scale datasets based on the pre-trained MAT are shown in Tab. 2. From these comparisons, we can draw the following observations:

Observation I: Simple graph structure information could deliver significant performance benefits. For instance, based on graph adjacency information, G-Adapters (S_1, S_2) obtain better results compared to G-Adapters (S_3, S_4) with distance-based structure information in Tab. 2. Besides, G-Adapter (S_1) also achieves the optimal performance in Tab. 1, we speculate that this might be because the graph adjacency matrix (S_1) contains complete structural information of the graph, that is, the distance-based structure (S_3) can be derived from S_1. In other word, this suggests that our model is not only remarkably expressive but also insensitive to graph structure. In the following statement, we take G-Adapter (S_1) as our baseline unless otherwise specified.

Observation II: Our proposed G-Adapter consistently outperforms traditional PEFTs and offers a better trade-off between the ratio of trainable parameters (γ) and model performance. Note that although BitFit updates the fewest number of parameters ($\gamma = 0.16\%$) on MolHIV and MolPCBA,

Method	Ratio* (γ)	RMSE (↓)		AUC (↑)				
		FreeSolv	ESOL	BBBP	Estrogen-α	Estrogen-β	MetStab$_{\text{low}}$	MetStab$_{\text{high}}$
Full FT	100%	0.286	0.270	0.764	0.979	0.778	0.863	0.878
Adapter	2.52%	0.327	0.320	0.724	0.978	0.768	0.846	0.859
Hyper.	2.43%	0.310	0.321	0.727	0.977	0.770	0.842	0.853
Comp.	1.56%	0.314	0.316	0.730	0.971	0.764	0.832	0.860
MAM	1.28%	0.302	0.292	0.743	0.980	0.776	0.851	0.872
LoRA	1.01%	0.309	0.284	0.726	0.979	0.781	0.839	0.878
BitFit	0.10%	0.321	0.314	0.739	0.977	0.770	0.848	0.805
Ours (S_1)	0.71%	**0.280**	**0.279**	0.750	0.976	**0.791**	0.865	**0.881**
Ours (S_2)	0.71%	0.282	0.286	**0.751**	**0.981**	0.788	**0.870**	0.874
Ours (S_3)	0.71%	0.291	0.289	0.744	0.973	0.786	0.860	0.861
Ours (S_4)	0.71%	0.298	0.282	0.747	0.975	0.775	0.858	0.869

Table 2: Comparison of PEFTs and full fine-tuning on small-scale datasets. The results are averaged from six seeds, where bold indicates the best result in PEFTs. Ratio* represents the mean ratio of trainable parameters over seven datasets.

Method	MolPCBA$_{(r=32)}$		MolHIV$_{(r=16)}$		FreeSolv$_{(r=8)}$		Estrogen-$\alpha_{(r=4)}$	
	Mem.	Infer	Mem.	Infer	Mem.	Infer	Mem.	Infer
Full FT	185	0.81	185	1.39	161	0.42	161	1.11
Adapter	5.0 (**180.0** ↓)	0.99	2.4 (**182.6** ↓)	1.46	1.1 (**159.9** ↓)	0.46	0.7 (**160.3** ↓)	1.15
LoRA	5.0 (**180.0** ↓)	0.97	2.4 (**182.6** ↓)	1.43	1.1 (**159.9** ↓)	0.47	0.7 (**160.3** ↓)	1.13
BitFit	0.3 (**184.7** ↓)	0.81	0.3 (**184.7** ↓)	1.39	0.2 (**160.8** ↓)	0.42	0.2 (**160.8** ↓)	1.11
Ours	2.9 (**182.1** ↓)	0.93	1.4 (**183.6** ↓)	1.41	0.7 (**160.3** ↓)	0.44	0.4 (**160.6** ↓)	1.12

Table 3: The comparison between PEFTs (Adapter, LoRA, BitFit, G-Adapter) and full fine-tuning in terms of inference speed (the millisecond per sample) and storage memory across various bottleneck sizes (r).

Method	GPU Mem.	Each Epoch
Full FT	21120	2.60
Ours (w/. Berg.)	10658 (**10462** ↓)	2.14 (**0.46** ↓)
Ours (w/o. Berg.)	8932 (**12188** ↓)	1.93 (**0.67** ↓)

Table 4: The comparison of GPU-memory usage and runtime per epoch under different settings during fine-tuning.

it yields the worst performances (0.709 AUC, 0.184 AP). In comparison, our proposed G-Adapter achieves 0.790 AUC, 0.269 AP with $\gamma = 0.24\%, \gamma = 1.89\%$, respectively, which is also the optimal solution compared to other PEFTs.

Observation III: G-Adapter could achieve competitive (even superior) performance on most datasets compared to full fine-tuning. For example, G-Adapter outperforms full fine-tuning by a significant margin (1.3% AUC) on Estrogen-β. We believe that the improvement on small-scale datasets is understandable: (i) full fine-tuning might meet serious over-fitting and catastrophic forgetting issues when the size of training set is small, while G-Adapter alleviates these concerns by merely updating specific blocks and freezing the original parameters; (ii) G-Adapter restricts the drastic updating of parameters via Bregman proximal point optimization strategy, which acts as a regularizer during fine-tuning and therefore boosts the generalization capacity.

Analysis

Efficiency of Training, Inference, and Memory Footprint

We investigate the following questions: (i) Does G-Adapter seriously affect the training and inference efficiency compared to full fine-tuning? (ii) Can G-Adapter offer significant benefits to GPU-memory usage and storage footprint? To begin with, in Fig. 5, we compare the training efficiency of full fine-tuning and PEFTs on datasets with different scales: MolPCBA (437K), MolHIV (41K), and Estrogen-α (2K). Then, the comparisons of inference speed and storage footprint are reported in Tab. 3. Last, in Tab. 4, we compare the GPU-memory usage and average runtime of a epoch between full fine-tuning and our method on MolHIV. It can be observed that: (i) although the introduced modules indeed affect the convergence in training and the inference speed, this effect would become negligible as the size of training data and bottleneck decrease; (ii) PEFTs significantly reduce the GPU-memory usage and storage footprint. For example, G-Adapter simply requires less than half of GPU-memory compared to full fine-tuning, since only the weights of specific modules are updated each time, which also slightly reduces the runtime during fine-tuning. On Estrogen-α ($r =$

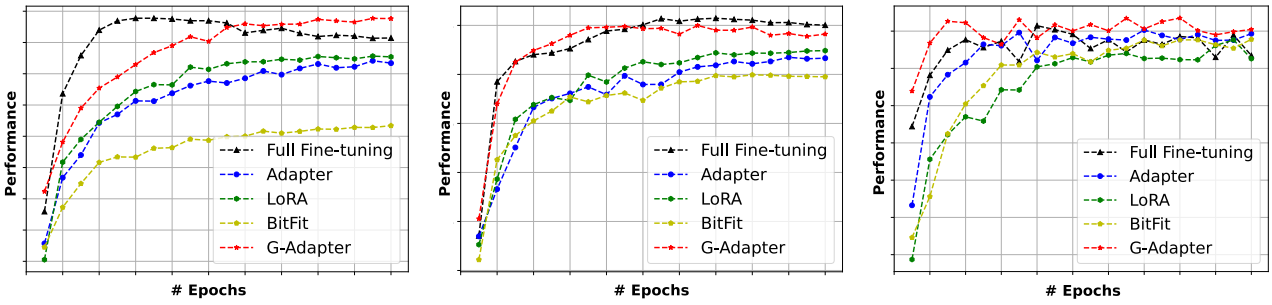


Figure 5: The comparison of training efficiency between PEFTs (Adapter, LoRA, BitFit, G-Adapter) and full fine-tuning.

Method	MolHIV	MolPCBA
G-Adapter (baseline)	**0.790**	**0.269**
G-Adapter (pre_mha)	0.752	0.246
G-Adapter (post_mha)	0.747	0.232
G-Adapter (pre_ffn)	0.781	0.258
G-Adapter (post_ffn)	0.770	0.260
G-Adapter (mha + ffn)	0.763	0.245
G-Adapter (w/o. S)	0.728	0.214
G-Adapter (w/o. pre_ln)	0.762	0.247
G-Adapter (w/o. post_ln)	0.755	0.250
G-Adapter (w/o. ln)	0.745	0.237
G-Adapter (w/o. act_fn)	0.766	0.240
G-Adapter (w/o. breg)	0.754	0.234

Table 5: The impact of insertion position and components.

4), full fine-tuning saves the complete checkpoint (161MB) for the task, while Adapter, BitFit, G-Adapter only need to save 0.7MB, 0.2MB, 0.4MB checkpoints, respectively.

The Impact of Insertion Position and Components

We discuss the impact of potential designs: (i) The insertion position. First, we insert G-Adapter block into the previous and posterior of MHA, denoted as **pre_mha** and **post_mha**, respectively. Then, G-Adapter block is also plugged into the previous and posterior of FFN, respectively, denoted as **pre_ffn** and **post_ffn**. Note that our baseline can be regarded as inserting G-Adapter block in the middle of FFN. Finally, similar to the insertion position of Adapter in Fig. 3, we insert two G-Adapter blocks into MHA and FFN, denoted as **mha + ffn**. (ii) Importance of each component. First, we remove the adjacency matrix (denoted as **w/o. S**) to explore the importance of graph structure information. Then, we separately remove the first, second, and both LayerNorm layers to explore their individual effects, denoted as **w/o. pre_ln**, **w/o. post_ln**, and **w/o. ln**. We also explore the role of the nonlinear activation function by removing it, denoted as **w/o. act_fn**. Last, the effect of Bregman proximal point optimization is evaluated by only using the vanilla loss function, denoted as **w/o. breg**. The experimental results in Tab. 5 demonstrate the effectiveness of our proposed methodology.

Method	MolHIV	MolPCBA	BBBP
Adapter	0.743	0.235	0.724
Adapter + S	0.749	0.242	0.733
Adapter + Breg.	0.747	0.239	0.731
Adapter + S + Breg.	0.747	0.245	0.742
LoRA	0.763	0.246	0.739
LoRA + S	0.766	0.252	0.742
LoRA + Breg.	0.765	0.249	0.743
LoRA + S + Breg.	0.771	0.253	0.746
G-Adapter (Ours)	**0.790**	**0.269**	**0.750**

Table 6: The impact of graph structure information and Bregman proximal point optimization on traditional PEFTs.

Can Graph Structure and Bregman Optimization Benefit Traditional PEFTs?

Considering that the main contributions of G-Adapter are the introduction of graph structure information and Bregman proximal point optimization, a natural question is: Can traditional PEFTs benefit from these technologies? To answer this question, we integrate graph structure information (S), Bregman proximal point optimization, and both technologies into Adapter and LoRA, respectively. The experiments are conducted on four datasets: MolHIV, MolPCBA, BBBP, and $\text{MetStab}_{\text{low}}$, and the results are reported in Tab. 6. It can be observed that the modified model has some performance improvement over the original model, but there is still a significant margin with our proposed G-Adapter, which further justifies the superiority of our designed PEFT architecture.

Conclusion

In this paper, we propose a novel structure-aware PEFT method, called G-Adapter, designed specifically for graph-based tasks utilizing pre-trained GTNs. Unlike the traditional PEFTs, which fail to achieve satisfactory performance due to the feature distribution shift issue, G-Adapter leverages graph structure information and Bregman proximal point optimization strategy to alleviate this concern. Extensive experiments on a variety of graph-based downstream tasks demonstrate the effectiveness of our proposed method.

Acknowledgments

This project is supported by the Young Scientists Fund of the National Natural Science Foundation of China (Grant No. 62006201).

References

Aghajanyan, A.; Gupta, S.; and Zettlemoyer, L. 2021. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 7319–7328.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization.

Ben-Zaken, E.; Goldberg, Y.; and Ravfogel, S. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 1–9.

Bo, D.; Shi, C.; Wang, L.; and Liao, R. 2023. Specformer: Spectral Graph Neural Networks Meet Transformers. In *The Eleventh International Conference on Learning Representations*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models Are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901.

Chen, J.; Zhang, A.; Shi, X.; Li, M.; Smola, A.; and Yang, D. 2023. Parameter-Efficient Fine-Tuning Design Spaces. In *The Eleventh International Conference on Learning Representations*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Diao, C.; and Loynd, R. 2023. Relational Attention: Generalizing Transformers for Graph-Structured Tasks. In *The Eleventh International Conference on Learning Representations*.

Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.-M.; Chen, W.; Yi, J.; Zhao, W.; Wang, X.; Liu, Z.; Zheng, H.-T.; Chen, J.; Liu, Y.; Tang, J.; Li, J.; and Sun, M. 2022. Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models.

Gaulton, A.; Bellis, L. J.; Bento, A. P.; Chambers, J.; Davies, M.; Hersey, A.; Light, Y.; McGlinchey, S.; Michalovich, D.; Al-Lazikani, B.; and Overington, J. P. 2012. ChEMBL: A Large-Scale Bioactivity Database for Drug Discovery. *Nucleic Acids Research*, 40(D1): D1100–D1107.

He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; and Neubig, G. 2022. Towards a Unified View of Parameter-Efficient Transfer Learning. In *International Conference on Learning Representations*.

Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; Laroussilhe, Q. D.; Gesmundo, A.; Attariyan, M.; and Gelly, S. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2790–2799.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2021. Open Graph Benchmark: Datasets for Machine Learning on Graphs.

Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Zhao, T. 2020. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2177–2190.

Jie, S.; and Deng, Z.-H. 2022. FacT: Factor-Tuning for Lightweight Adaptation on Vision Transformer.

Jin, B.; Zhang, Y.; Meng, Y.; and Han, J. 2023. Edgeformers: Graph-Empowered Transformers for Representation Learning on Textual-Edge Networks. In *The Eleventh International Conference on Learning Representations*.

Karimi Mahabadi, R.; Ruder, S.; Dehghani, M.; and Henderson, J. 2021. Parameter-Efficient Multi-task Fine-tuning for Transformers via Shared Hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 565–576.

Khoo, L. M. S.; Chieu, H. L.; Qian, Z.; and Jiang, J. 2020. Interpretable Rumor Detection in Microblogs by Attending to User Interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8783–8790.

Kim, J.; Nguyen, D. T.; Min, S.; Cho, S.; Lee, M.; Lee, H.; and Hong, S. 2022. Pure Transformers Are Powerful Graph Learners. In *Advances in Neural Information Processing Systems*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; Létourneau, V.; and Tossou, P. 2022. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 4582–4597.

Lin, K.; Wang, L.; and Liu, Z. 2021. Mesh Graphormer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 12919–12928. ISBN 978-1-66542-812-5.

Liu, X.; Ji, K.; Fu, Y.; Tam, W.; Du, Z.; Yang, Z.; and Tang, J. 2022. P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 61–68.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Mahabadi, R. K.; Henderson, J.; and Ruder, S. 2021. Compacter: Efficient Low-Rank Hypercomplex Adapter Layers. In *Advances in Neural Information Processing Systems*.

Maziarka, Ł.; Danel, T.; Mucha, S.; Rataj, K.; Tabor, J.; and Jastrzebski, S. 2020. Molecule Attention Transformer.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. GraphiT: Encoding Graph Structure in Transformers.

Min, E.; Chen, R.; Bian, Y.; Xu, T.; Zhao, K.; Huang, W.; Zhao, P.; Huang, J.; Ananiadou, S.; and Rong, Y. 2022. Transformer for Graphs: An Overview from Architecture Perspective.

Podlewska, S.; and Kafel, R. 2018. MetStabOn—Online Platform for Metabolic Stability Predictions. *International Journal of Molecular Sciences*, 19(4): 1040.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Rücklé, A.; Geigle, G.; Glockner, M.; Beck, T.; Pfeiffer, J.; Reimers, N.; and Gurevych, I. 2021. AdapterDrop: On the Efficiency of Adapters in Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 7930–7946.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30.

Wang, R.; Tang, D.; Duan, N.; Wei, Z.; Huang, X.; Ji, J.; Cao, G.; Jiang, D.; and Zhou, M. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 1405–1418.

Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; and Pande, V. 2018. MoleculeNet: A Benchmark for Molecular Machine Learning. *Chemical Science*, 9(2): 513–530.

Xia, J.; Zhu, Y.; Du, Y.; and Li, S. Z. 2022. A Survey of Pretraining on Graphs: Taxonomy, Methods, and Applications.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do Transformers Really Perform Badly for Graph Representation? In *Advances in Neural Information Processing Systems*.

Yu, B. X. B.; Chang, J.; Liu, L.; Tian, Q.; and Chen, C. W. 2022. Towards a Unified View on Visual Parameter-Efficient Transfer Learning.

Yuan, W.; Gu, X.; Li, H.; Dong, Z.; and Zhu, S. 2023. Monocular Scene Reconstruction with 3D SDF Transformers. In *The Eleventh International Conference on Learning Representations*.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020. Graph-Bert: Only Attention Is Needed for Learning Graph Representations.

Zhang, Q.; Chen, M.; Bukharin, A.; He, P.; Cheng, Y.; Chen, W.; and Zhao, T. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*.

Zhao, J.; Li, C.; Wen, Q.; Wang, Y.; Liu, Y.; Sun, H.; Xie, X.; and Ye, Y. 2021. Gophormer: Ego-Graph Transformer for Node Classification.